Descriptive statistics with R

María-Eglée Pérez

Biostatistics, Epidemiology and Bioinformatics Core (BEBIC)
UPR-MDACC Partnership for Excellence in Cancer Research
NIH NCI U54 GRANT 2U54CA096297-11

Contents

Interfaces for R

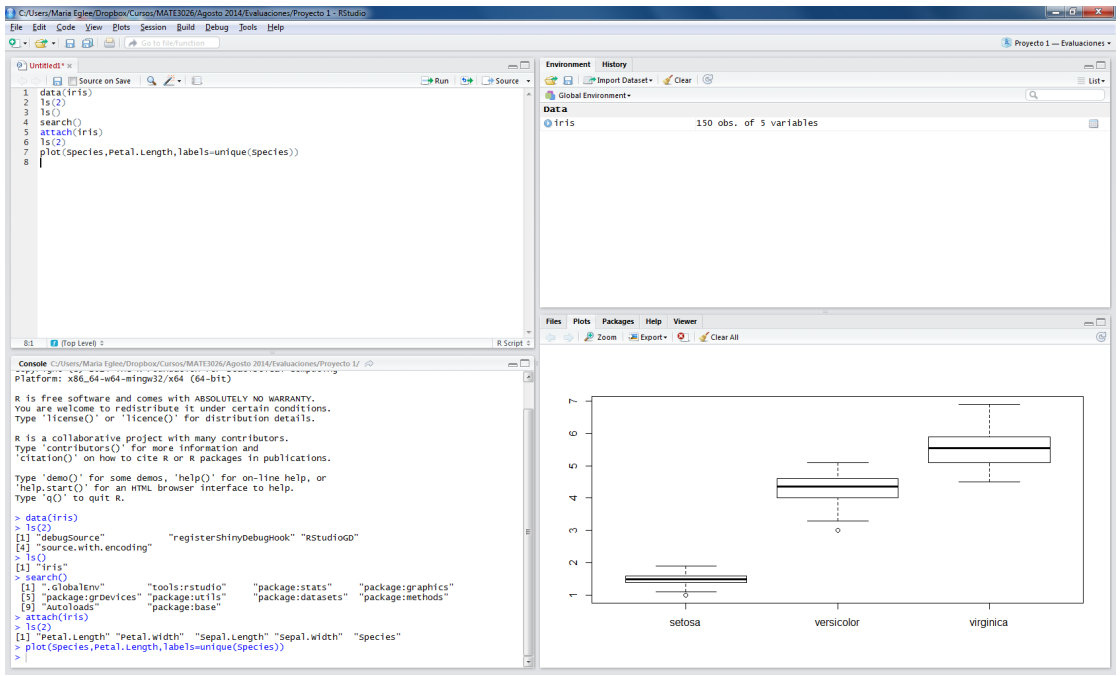
Even though R can be used directly, many users prefer using an interface. Some common interfaces are:

- RStudio: is an "Integrated Development Environment" for R (`http://www.r-studio.com`)
- RCommander: a Graphic User Interface (GUI) for R, which incorporates point and click menus for many statistical procedures. It can be integrated with RStudio (`http://www.rcommander.com`)
- Deducer: another GUI for R (`http://www.deducer.org`).

RStudio Window

Text Editor

Workspace



Console

Graphs

Extending R functionalities: Packages

One of the biggest advantages of R is its wide community of users and collaborators. Members of this community produce extensions to R, or *packages*.

A list of existing packages can be found at `http://cran.r-project.org`. On March 19, 2015, there were 6424 package.

Packages can be installed:

- with command `install.packages("name")`;
- using the menu item `Packages > Install packages` in R (R for windows);
- using the tab "packages" in the graph region of the RStudio window;

If we are going to use a package during the actual session, we can load it using `require(`*name*`)` or `library(`*name*`)`

We can see which packages are loaded using `search()` (which returns the search list) or `sessionInfo()`

Data Frames

Usually, we have a set of observations for each subjects in the study. A *data frame* is a structure with matrix shape, in which different data columns (numeric, character or logical) can be stored. In general, rows correspond to different subjects, and columns correspond to variables observed for each subject.
Names can be assigned to the rows and columns of a data frame.
As an example, consider the data frame `mtcars`, which is available in R

```
> data(mtcars)
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Using data in data frames

Data contained in data frames can be used in several ways.

- Attaching the data frame. This includes the data frame in the search list, so the columns in it can be identified as variables.

```
> attach(mtcars)
> search()
 [1] ".GlobalEnv"        "mtcars"            "tools:rstudio"
 [4] "package:stats"     "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"  "package:methods"
[10] "Autoloads"         "package:base"
> objects(2)
 [1] "am"   "carb" "cyl"  "disp" "drat" "gear" "hp"   "mpg"
 [9] "qsec" "vs"   "wt"
> summary(mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
```

Once the data frame is not needed anymore, it can be detached

```
> detach(mtcars)
> search()
 [1] ".GlobalEnv"         "tools:rstudio"     "package:stats"
 [4] "package:graphics"   "package:grDevices" "package:utils"
 [7] "package:datasets"   "package:methods"   "Autoloads"
[10] "package:base"
> summary(mpg)
Error in summary(mpg) : object 'mpg' not found
```

- Using `with`

```
> with(mtcars,summary(mpg))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
```

Modelling functions have options for specifying the data frame where data are.

Working directory

R will read and write information from a *working directory*. Working directory can be changed during the session.

```
> # Get actual working directory
> getwd()
[1] "C:/Users/Maria Eglee/Dropbox/Cursos/MATE3026/Agosto 2014/Evaluaciones"
> # Set new working directory
> setwd("C:/Users/Maria Eglee//Dropbox//Grants//U54//R workshop//Slides")
> getwd()
[1] "C:/Users/Maria Eglee/Dropbox/Grants/U54/R workshop/Slides"
```

Working directory can also be set using menus, both in R and in RStudio

Reading data from delimited files

Text files can be read using `read.table`.
The default separator for `read.table` is a "blank space" (any number of spaces or tabs).
We will use a dataset included in the Agresti and Franklin (2012) CD, containing data on human development (Program of the United Nations for Development, PNUD)

```
> HumanDevelopTxt=read.table("./Datasets/human_development.txt",
+ header=T,sep="")
 Error in scan(file, what, nmax, sep, dec, quote, skip, nlines,
 na.strings,  : line 24 did not have 7 elements
```

This happens because some country names have two or more words!

To solve this, the separator can be substituted by a tab.

```
> HumanDevelopTxt=read.table("./Datasets//human_development.txt",
+ header=T,sep="\t")
> class(HumanDevelopTxt)
[1] "data.frame"
> head(HumanDevelopTxt,5)
       C1.T INTERNET   GDP  CO2 CELLULAR FERTILITY LITERACY
1   Algeria     0.65  6.09  3.0      0.3       2.8     58.3
2 Argentina    10.08 11.32  3.8     19.3       2.4     96.9
3 Australia    37.14 25.37 18.2     57.4       1.7    100.0
4   Austria    38.70 26.73  7.6     81.7       1.3    100.0
5   Belgium    31.04 25.52 10.2     74.7       1.7    100.0

> HumanDevelopTxt[24,]
          C1.T INTERNET   GDP CO2 CELLULAR FERTILITY LITERACY
24 New Zealand    46.12 19.16 8.1     59.9         2      100
```

One of the variants of `read.table`, `read.delim`, already includes this change and expects column names by default.

```
> HumanDevelopTxt=read.delim("./Datasets/human_development.txt")
> head(HumanDevelopTxt)
      C1.T INTERNET   GDP  CO2 CELLULAR FERTILITY LITERACY
1  Algeria     0.65  6.09  3.0      0.3       2.8     58.3
2 Argentina    10.08 11.32  3.8     19.3       2.4     96.9
3 Australia    37.14 25.37 18.2     57.4       1.7    100.0
4  Austria     38.70 26.73  7.6     81.7       1.3    100.0
5  Belgium     31.04 25.52 10.2     74.7       1.7    100.0
6   Brazil      4.66  7.36  1.8     16.7       2.2     87.2
```

Another variant of `read.table`, `read.csv`, allows to read comma separated values files (which can be produced from Excel)
This is usually the easiest way of getting data into R (personal opinion!).

```
> HumanDevelopCsv=read.csv("./Datasets/human_development.csv")
> head(HumanDevelopCsv,5)
       C1.T INTERNET   GDP  CO2 CELLULAR FERTILITY LITERACY
1   Algeria     0.65  6.09  3.0      0.3       2.8     58.3
2 Argentina    10.08 11.32  3.8     19.3       2.4     96.9
3 Australia    37.14 25.37 18.2     57.4       1.7    100.0
4   Austria    38.70 26.73  7.6     81.7       1.3    100.0
5   Belgium    31.04 25.52 10.2     74.7       1.7    100.0
```

The package `foreign` in R contains functions for reading and writing data stored by some versions of Minitab, SAS, SPSS, Stata (frozen at version 12, but your can use `saveold` in Stata) and other statistical software

Describing Categorical Data

We will use the data set `survey` in library MASS. According to help page, "this data frame contains the responses of 237 Statistics I students at the University of Adelaide to a number of questions". Data include sex, age, height, frequency of smoking, frequency of exercising and others.

```
> require(MASS)
> data(survey)
> names(survey)
> attach(survey)
```

Lets consider the frequency of smoking

```
> summary(Smoke)
Heavy Never Occas Regul  NA's
   11   189    19    17     1
```

First thing we note is that by default R orders the levels of a factor alphabetically.

We will assign a more natural order redefining the variable

```
> Smoke=factor(Smoke, levels=c("Never","Occas","Regul","Heavy"))
> table(Smoke)
Smoke
Never Occas Regul Heavy
  189    19    17    11
```

The object in the data frame `survey` is not changed.

We can also tabulate the data on frequency of exercise in the variable Exer, and redefine it with a more natural order in the levels.

```
> table(Exer)
Exer
Freq None Some
 115   24   98
> Exer=factor(Exer,levels=c("Freq","Some", "None"))
> table(Exer)
Exer
Freq Some None
 115   98   24
```

For exploring possible associations between frequency of exercise and frequency of smoking, a contingency table can be generated.

```
> ExerSmoke.tab=table(Exer,Smoke)
> ExerSmoke.tab
       Smoke
Exer    Never Occas Regul Heavy
  Freq     87    12     9     7
  Some     84     4     7     3
  None     18     3     1     1
```

Totals by rows and columns can be obtained using the command `apply`

```
> # Totals by row
> apply(ExerSmoke.tab,1,sum)
Freq Some None
 115   98   23
> # Totals by column
> apply(ExerSmoke.tab,2,sum)
Never Occas Regul Heavy
  189    19    17    11
> # Total of observations
> sum(ExerSmoke.tab)
[1] 236
```

Tables of proportions can be obtained. For example, lets get a table of proportions by rows.

```
> ExerSmoke.prop=ExerSmoke.tab/apply(ExerSmoke.tab,1,sum)
> round(ExerSmoke.prop,3)
       Smoke
Exer    Never Occas Regul Heavy
  Freq 0.757 0.104 0.078 0.061
  Some 0.857 0.041 0.071 0.031
  None 0.783 0.130 0.043 0.043
```

Results are rounded to three decimal places for display.

Barplots

For producing a barplot of the frequency of smoking, we can use the command `barplot`.

```
> barplot(table(Smoke))
```

This graph can be formatted using some graphical options

```
> barplot(table(Smoke),names=c("Never","Ocassionally", "Regularly",
+  "Heavily"),  xlab="Frequency of smoking", ylab="No. of responses",
+ cex.names=0.8,col="lightblue")
```

`colors()` gives a list of all available colors.

Barplots can also be drawn horizontally. Order of classes can be reverted.

```
> barplot(rev(table(Smoke)),names=rev(c("Never","Ocassionally",
+ "Regularly", "Heavily")), ylab="Frequency of smoking", cex.names=0.7,
+ col="lightblue",horiz=T)
```

A side by side barplot can be used for displaying a contingency table

```
> barplot(ExerSmoke.tab,beside=T,xlab="Frequency of smoking",
+ col=terrain.colors(3))
> legend(locator(1),fill=terrain.colors(3),legend=c("None","Some",
+ "Frequent"), title="Frequency of excercise")
```

The groups present similar behavior.
Same graph in grayscale and locating the legend in the top right angle

```
> barplot(ExerSmoke.tab,beside=T,xlab="Frequency of smoking",
+ col=gray(seq(0,1,len=3)))
> legend("topright",fill=gray(seq(0,1,len=3)),legend=c("None","Some",
+ "Frequent"), title="Frequency of excercise")
```

Mosaic Plots

Mosaic plots are a visual tool for displaying contingency tables (and log linear models). Each "tile" corresponds to a cell in the table. (Hartigan and Kleiner, 1981).

```
> mosaicplot(ExerSmoke.tab,col=terrain.colors(4),
+ main=" Exercise frequency vs Smoking frequency",
+ las=1,xlab="Excercise")
```

Graphical option `las` manages the orientation of labels with respect to the axis (0 = always parallel to the axis (default option), 1= always horizontal, 2= always perpendicular to the axis, 3=always vertical).

Exploring Quantitative Variables

File `latam2011.csv` contains demographical data for Latin American countries (obtained from the website of the Panamerican Health Organisation, PAHO). The variables in this dataset are

1. Total population (in thousands)
2. Population older than 60 years (percentage).
3. Birth rate ($\times$ 1,000 pop.)
4. Death rate ($\times$ 1,000 pop.)
5. Life expectancy at birth (years)

All these variables are quantitative.

Dot charts

For a small quantitative data set, a dot chart (Cleveland, 1985) can be useful. Dot charts are similar to bar plots, but the have been designed for easier visualisation and lower use of ink.

```
> latam.frm <- read.csv("latam2011.csv")
> attach(latam.frm)
> dotchart(TotalPop,labels=rownames(latam.frm),
+ main="Population for Latin America \n Year 2011")
```

Data can be sorted for better display.

```
> dotchart(sort(TotalPop),
+ labels=rownames(latam.frm)[order(TotalPop)],
+ main="Population for Latin America \n Year 2011")
```

Strip chart

A *strip chart* or *dotplot* plots values along a line. If the option `method="stack"` is added, repeated values are displayed stacking dots.

```
> stripchart(TotalPop,xlab="Total Population (thousands)", pch=16)
> stripchart(round(BirthRate,0), xlab="BirthRate (x1,000)", pch=16)
> stripchart(round(BirthRate,0),method="stack",
+ xlab="BirthRate (x1,000)", pch=16)
```

Strip charts can also be useful to compare two or more sets of data. Suppose, for example, that we want to compare the birth rates of the countries in South America with countries in Central America and the Caribbean.

```
> SouthAm=c(1,0,1,1,1,1,0,0,0,1,0,0,0,0,0,0,1,1,0,1,1)
> stripchart(round(BirthRate,0)~SouthAm,method="stack",
+ xlab="Birth Rate (x1,000)")
> text(locator(),c("South America",
+ "Central America and Caribbean"))
```

Histogram

Data are grouped in classes, and they are displayed through a bar plot, in which the area of each bar is proportional to the frequency of the class. Usually, all classes have the same width, and the height of the bars is proportional to the frequency of the class
The following command can be used to obtain a histogram in *R*

```
hist(x, breaks = "Sturges", freq = TRUE,
     right = TRUE, ...)
```

`breaks` can take one of the following types of value

- A vector giving the breakpoints between histogram cells
- A single number giving the number of cells for the histogram
- A character string naming an algorithm to compute de number of cells `"Sturges"`. `"Scott"` or `"FD"` (Freedman-Diaconis algorithm).
- A function to compute the number of cells.

In the last three cases the number is a suggestion only.

`freq` is a logical value; if `TRUE`, the histogram graphic is a representation of frequencies; if `FALSE`, probability densities are plotted (so that the histogram has a total area of one).

`right` is a logical value; if `TRUE`, the histograms cells are right-closed (left open) intervals.

Other general options like `xlim`, `ylim`, `main`, etc. can be used

The algorithms for deciding the number of classes of the histogram are:

1. **Sturges' rule:** The number of classes k is calculated as

$$k = 1 + \log_2 n = 1 + 3.34 \log_{10} n$$

2. **Scott's rule:** The class width h is

$$h = 3.5\, s\, n^{-\frac{1}{3}}$$

where s is the standard deviation of the sample

3. **Freedman and Diaconis:** The class width h is

$$h = 2(IQ)n^{-\frac{1}{3}}$$

where IQ is the interquartile range of the sample.

Most statistical packages use Sturges' rule (or an extension of it) for selecting the number of classes when constructing a histogram. Sturges' rule is also widely recommended in introductory statistics textbooks. Nevertheless, it is known that Sturges' rule leads to oversmoothed histograms, specially for large n or skewed data.

To see the effect of the change in the rule for defining the clases, consider these two histograms

```
> hist(TotalPop,main="Total Population")
> hist(TotalPop,breaks="FD",main="Total Population")
```

These data are skewed, so Freedman-Diaconis algorithm performs better than Sturges one.

The `hist` command can be used without plotting the graph using the option `plot=F`. In that case, the output is a list containing the following objects

- `breaks:` the k+1 cell boundaries (= 'breaks' if that was a vector).
- `counts:` k integers giving the number of observations inside each class.
- `density:` estimated density values. If the bases of the classes are equal to 1, they are the relative frequencies "counts/n" . In general the values satisfy that the area of the histogram is 1.

- `mids:` the k cell midpoints.
- `xname:` a character string with the actual 'x' argument name.
- `equidist:` logical, indicating if the distances between breaks are all the same.

```
> lehist = hist(LifeExp,plot=F)
> lehist
$breaks
[1] 66 68 70 72 74 76 78 80

$counts
[1] 1 0 1 6 5 4 4

$density
[1] 0.02380952 0.00000000 0.02380952 0.14285714 0.11904762
[6] 0.09523810 0.09523810

$mids
[1] 67 69 71 73 75 77 79
```

```
$xname
[1] "LifeExp"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
```

This results could be used, for example, to add a cumulative frequency curve to the histogram.

```
> sum(lehist$counts)
[1] 21
> plot(lehist,ylim=c(0,21),xlab="Life Expectancy (Years)",
+ main="Life Expectancy \n Latin America 2011",col="gray")
> lines(lehist$breaks,c(0,cumsum(lehist$counts)))
```

Observations:

1. `plot` is the fundamental command for graphics in *R*, and its behavior changes with the class of the object to which it is being applied.
2. `lines` and `points` add lines and points to a graph previously plotted. This is just an example on how *R* allows to interact with graphics.
3. `cumsum` calculates the cumulative sum of the elements of a vector.

Numerical Descriptive Measures

- **Centrality measures**
  - **Mean:**

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

  - **Median:** is the value such that 50% of the ordered data is below it and 50% is over it.

The mean uses all the information in the data, but is very sensitive to skewness and to the presence of extreme data. The median uses less information (only the order of the data and the central observations), but it is not altered if one observation (or a small subset of observations) contains large errors of measurement or transcription. Large differences between the mean and the median indicate either skewness or presence of outliers.

  - **Trimmed mean:** A certain percentage of observations is removed in both extremes of the data.

In *R*, the mean and the trimmed mean can be calculated using the command `mean(x,trim=0)`, where `trim` is the proportion of observations to be removed, and can take values from 0 (default value, corresponding to the mean) to 0.5 (in that case, we obtain the median).
The median can be obtained using the command `median(x)`.

```
> mean(TotalPop)
[1] 27534.76
> median(TotalPop)
[1] 10088
> mean(TotalPop,trim=0.05)
[1] 20066.16
> mean(TotalPop,trim=0.1)
[1] 15475.53
```

The large difference between the mean and the median corresponds to our prior observation of the existence of two very large observations, corresponding to Mexico and Brazil.

- **Dispersion measures**
  - **Range:**

$$\text{Range} = X_{\text{max}} - X_{\text{min}}$$

  - **Interquartile range:**
    The quartiles q_1, q_2 y q_3 of a certain dataset divide the ordered data in four parts which contain the same number of observations (Note that q_2 =median).

$$\text{Interquartile Range} = q_3 - q_1$$

  - **Sample variance:**

$$S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

    The *standard deviation* is defined as:

$$S = \sqrt{S^2}$$

We have already used `max` y `min` to obtain the maximum and minimum values of a vector. The command `quantile` can be used to obtain any sample quantile, and in particular the maximum, the minimum and the quartiles. Its general syntaxis is

```
quantile(x,probs=seq(0, 1, 0.25),na.rm=FALSE,names=TRUE)
```

The algorithm for the calculation of quantiles can be changed using the option `type` (default value is `type = 7`)

The variance is calculated using `var`. The standard deviation can be obtained as the square root of the variance or using the command `sd`.

```
> quantile(TotalPop)
    0%    25%    50%    75%   100%
   318   5870  10088  29400 196655
> quantile(TotalPop,.75)-quantile(TotalPop,.25)
  75%
23530
> sd(TotalPop)
[1] 46271.94
```

The central 50% of the data is in an interval of length 23530, and the "typical" deviation from the mean is more than 46 thousand. This is due to the presence of very large observations.

The command summary calculates these descriptive quantities

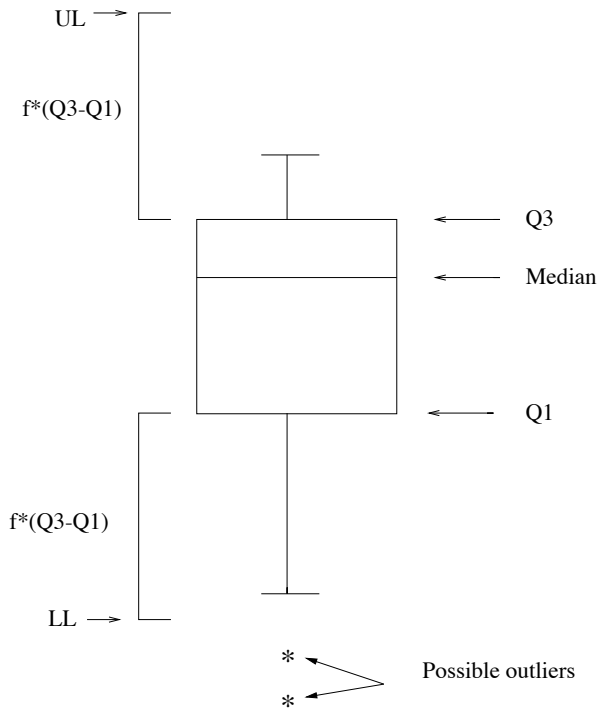
```
> summary(TotalPop)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    318    5870   10090   27530   29400  196700
```

summary can also be applied to a data frame

```
> summary(latam.frm[,1:4])
    TotalPop          PopOT60        BirthRate       DeathRate
 Min.   :   318   Min.   : 5.70   Min.   : 9.8   Min.   :3.800
 1st Qu.:  5870   1st Qu.: 7.80   1st Qu.:15.5   1st Qu.:5.000
 Median : 10088   Median : 9.10   Median :19.9   Median :5.500
 Mean   : 27535   Mean   :10.29   Mean   :19.7   Mean   :5.876
 3rd Qu.: 29400   3rd Qu.:10.60   3rd Qu.:23.2   3rd Qu.:6.500
 Max.   :196655   Max.   :18.40   Max.   :31.8   Max.   :9.300
```

Box and Whiskers plot

A box and whiskers plot summarizes several numerical descriptive measures: minimum, maximum and quartiles. Characteristics of the data such as symmetry and possible outliers can be observed in a box plot.
Structure of a *box plot*:



The whiskers end in the last observation inside the admissible limits $LL = q1 - f(q_3 - q_1)$ and $UL = q3 + f(q_3 - q_1)$.

The command in *R* for drawing a box plot is `boxplot(x,...)`. The default value of f is 1.5, but it can be changed using the option `range`.

```
> boxplot(TotalPop,ylab=Total Population (Thousands)")
```

Two or more box plots can be drawn in the same plot for comparison

```
> boxplot(BirthRate,DeathRate,names=c("Birth Rate","Death Rate"))
```

Suppose now we want to compare different levels of a factor. For example, in the `survey` dataset, we want to compare pulse rates for different levels of exercise.

```
> class(Pulse)
[1] "integer"
> class(Exer)
[1] "factor"
```

`split` creates a list that contains all the values of a variable splitted according to the value of a factor. `lapply` applies a function to all elements in a list.

```
> lapply(split(Pulse,Exer),summary)
$Freq
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  40.00   65.00   71.00   71.97   78.00  104.00      20

$Some
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  35.00   69.50   76.00   76.19   84.25  100.00      18

$None
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  50.00   68.00   76.00   76.76   86.00  104.00       7
```

Plotting a factor vs a quantitative variables produces box plots of the quantitative variable for each level of the factor.

```
> plot(Exer,Pulse)
> plot(Exer,Pulse,notch=T)
```

Notches are used to compare the medians. According to Chambers *et al* (1983) the absence of overlapping in two boxes' notches is "strong evidence" their medians differ

Exploring Relationships Between Quantitative Variables

`plot` can also be used for drawing scatterplots for exploring the relationship between two variables.

```
> plot(LifeExp,BirthRate,xlab="Life Expectancy (Years)",
+ ylab="Birth Rate (x 1000pop)")
```

The option `pch` changes the appearance of the plotting character. Plotting characters can be indicated by numbers or by the character itself.

```
> plot(LifeExp,BirthRate,xlab="Life Expectancy (Years)",
+ ylab="Birth Rate (x 1000pop)",pch=16)
> plot(LifeExp,BirthRate,xlab="Life Expectancy (Years)",
+ ylab="Birth Rate (x 1000pop)",pch="X")
```

The color of the points in the graphic can be changed using `col`.

The command `pairs` produces a matrix of scatterplots for comparing all the columns in a matrix or data frame.

```
> pairs(latam.frm)
```

From this plot several features of the data can be seen.

Correlations between the variables can be found using `cor`. The command can be applied to a pair of vectors or to a data frame or matrix.

```
> round(cor(latam.frm),3)
           TotalPop PopOT60 BirthRate DeathRate LifeExp
TotalPop      1.000  -0.009    -0.209     0.006  -0.039
PopOT60      -0.009   1.000    -0.834     0.797   0.628
BirthRate    -0.209  -0.834     1.000    -0.465  -0.760
DeathRate     0.006   0.797    -0.465     1.000   0.044
LifeExp      -0.039   0.628    -0.760     0.044   1.000
```

For identifying the extreme point in the scatterplot of death rate vs life expectancy, lets draw this graph separately

```
> plot(LifeExp,DeathRate,xlab="Life Expectancy (Years)"
+ ,ylab="Death Rate (x 1000 pop.)")
> identify(LifeExp,DeathRate,labels=rownames(latam.frm))
[1] 7
```

The command `identify` reads the position of the graphics pointer when the (first) mouse button is pressed. It then searches the coordinates for the point closest to the pointer. If this point is close enough to the pointer, its index will be returned as part of the value of the call. It is stopped with the right button of the mouse or when all points are identified.
The extreme point at the lleft corresponds to Bolivia.

Another important option for `plot` is `type`. Some possible types are

`type=''p''`	Points (default option).
`type=''l''`	Lines.
`type=''b''`	Both lines and points.
`type=''c''`	The lines part alone of "b".
`type=''o''`	Points and lines overploted.
`type=''n''`	No plotting.

More options can be found in the online help of *R*.

As an example, lets draw a graph of Death Rate vs Life Expectancy in which we are going to highlight data corresponding to the following countries: Bolivia, Colombia, Puerto Rico, Uruguay and Venezuela. First, we will create a logical indicator vector for the countries we are interested in:

```
> countries=rownames(latam.frm)
> selec=countries=="Bolivia"|countries=="Colombia"|
+ countries=="PuertoRico"|countries=="Uruguay"|
+ countries=="Venezuela"
> selec
 [1] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
 [9] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[17] FALSE FALSE  TRUE  TRUE  TRUE
```

Note the use of | as a logical "or".

Next, we will make an empty plot using all the data. This guarantees that we will be able of drawing all the points in the graph.

```
>plot(DeathRate,LifeExp,xlab="Death Rate (x1000)",
+ ylab="Life Expectancy (years)",type="n")
```

Next, we add to the plot the countries of interest using red filled circles

```
> points(DeathRate[selec],LifeExp[selec],pch=16,col="red")
```

Now, we add the rest of the points in the graph using black filled circles.

```
> points(DeathRate[!selec],LifeExp[!selec],pch=16)
```

Note the use of `!` as a logical "not".
Finally, we add labels to the countries of interest. The size of the text is going to be reduced to 80% of the default size.

```
> identify(DeathRate[selec],LifeExp[selec],pch=16,
+ col="red",labels=countries[selec])
[1] 1 2 3 4 5
```